

UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPPPP	
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPPPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU	EEEEEEEEEEEEEEEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU	EEEEEEEEEEEEEEEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU	EEEEEEEEEEEEEEEE	TTT	PPP	


```
SSSSSSSS  AAAAAA  TTTTTTTTTT  SSSSSSSS  SSSSSSSS  FFFFFFFF  11  888888
SSSSSSSS  AAAAAA  TTTTTTTTTT  SSSSSSSS  SSSSSSSS  FFFFFFFF  11  888888
SS      AA      AA      TT      TT      FF      1111  88      88
SS      AA      AA      TT      TT      FF      1111  88      88
SS      AA      AA      TT      TT      FF      11    88      88
SS      AA      AA      TT      TT      FF      11    88      88
SSSSSS  AA      AA      TT      TT      FFFFFFFF  11  888888
SSSSSS  AA      AA      TT      TT      FFFFFFFF  11  888888
SS      AAAAAAAAAA  TT      TT      FF      11  88      88
SS      AAAAAAAAAA  TT      TT      FF      11  88      88
SS      AA      AA      TT      TT      FF      11  88      88
SS      AA      AA      TT      TT      FF      11  88      88
SSSSSS  AA      AA      TT      TT      FF      111111  888888
SSSSSS  AA      AA      TT      TT      FF      111111  888888
                                     ....
                                     ....
                                     ....
                                     ....

LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS
```


(1)	65	DECLARATIONS
(1)	89	OWN STORAGE
(1)	167	R/W PSECT
(1)	264	SATSSF18
(2)	319	CREPRC TESTS
(2)	506	SETPRV TESTS
(2)	551	UNWIND TESTS
(2)	628	REG_SAVE
(2)	649	REG_CHECK
(2)	692	PRINT_FAIL
(2)	728	MOD_MSG_PRINT
(2)	741	CHMRTN


```
0000 1 .TITLE SATSSF18 - SATS SYSTEM SERVICE TESTS (FAILING S.C.)
0000 2 .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 * ALL RIGHTS RESERVED.
0000 10
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 * TRANSFERRED.
0000 17
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 * CORPORATION.
0000 21
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24
0000 25 *****
0000 26
0000 27
0000 28
0000 29 ++
0000 30 FACILITY: SATS SYSTEM SERVICE TESTS
0000 31
0000 32 ABSTRACT: The SATSSF18 module tests the execution of the following
0000 33 VMS system services, invoked in such a way as to expect failing
0000 34 status codes:
0000 35 $CREPRC
0000 36 $SETPRV
0000 37 $UNWIND
0000 38
0000 39
0000 40 ENVIRONMENT: User mode image; needs CMKRNL privilege,
0000 41 dynamically acquires other privileges, as needed.
0000 42
0000 43 AUTHOR: Larry D. Jones, CREATION DATE: NOVEMBER, 1979
0000 44
0000 45 MODIFIED BY:
0000 46
0000 47 V03-005 LDJ0005 Larry D. Jones, 23-Jul-1984
0000 48 Modified for addition of one new status flag.
0000 49
0000 50 V03-004 LDJ0004 Larry D. Jones, 19-Apr-1984
0000 51 Modified for addition of one new status flag. Fixed
0000 52 duplicate process name failure.
0000 53
0000 54 V03-003 LDJ0003 Larry D. Jones, 25-Mar-1983
0000 55 Modified for addition of three new status flags.
0000 56
0000 57 V03-002 LDJ0002 Larry D. Jones, 07-Aug-1981
```


SATSSF18
V04-000

- SATS SYSTEM SERVICE TESTS (FAILING S. ^{G 2} 16-SEP-1984 01:42:11 VAX/VMS Macro V04-00
5-SEP-1984 04:22:29 [UETP.SRC]SATSSF18.MAR;1

Page 2
(1)

0000 58 :
0000 59 :
0000 60 :
0000 61 :
0000 62 :**
0000 63 :--

Modified for addition of disable WS adjust status flag.

V03-001 LDJ0001 Larry D. Jones, 17-Sep-1980
Modified to conform to new build command procedures.


```

0000 65 .SBTTL DECLARATIONS
0000 66 :
0000 67 : MACRO LIBRARY CALLS
0000 68 :
0000 69 $CHFDEF ; condition handler frame offsets
0000 70 $JPIDEF ; GETJPI definitions
0000 71 $PQLDEF ; process quota list definitions
0000 72 $PRVDEF ; privilege definitions
0000 73 $UETPDEF ; UETP message definitions
0000 74 $SFDEF ; stack frame offset definitions
0000 75 $SHR MESSAGES UETP,116,<<TEXT,INFO>> ; UETP$ TEXT definition
0000 76 $SSDEF ; SS definitions
0000 77 $STSDEF ; STS definitions
0000 78 :
0000 79 : Equated symbols
0000 80 :
00000000 0000 81 WARNING = 0 ; warning severity value for msgs
00000001 0000 82 SUCCESS = 1 ; success
00000002 0000 83 ERROR = 2 ; error
00000003 0000 84 INFO = 3 ; information
00000004 0000 85 SEVERE = 4 ; fatal
00000001 0000 86 PRVHND_SXV40 = 1 ; page 0 address for SETEXV
0000 87

```



```
0000 89 .SBTTL OWN STORAGE
00000000 90 .PSECT RODATA,RD,NOWRT,NOEXE,LONG
0000 91
0000 92 TEST_MOD_NAME:
0000 93 .ASCIC /SATSSF18/ ; needed for SATSMS message
38 31 46 53 53 54 41 53 00' 0000
08 0000
0009 94 TEST_MOD_NAME D:
0009 95 .ASCIC /SATSSF18/ ; module name
46 53 53 54 41 53 00000011'010E0000'
38 31 0017
0019 96 TEST_MOD_BEGIN:
0019 97 .ASCIC /begin/
6E 69 67 65 62 00' 0019
05 0019
001F 98 TEST_MOD_SUCC:
001F 99 .ASCIC /successful/
6C 75 66 73 73 65 63 63 75 73 00' 001F
0A 001F
002A 100 TEST_MOD_FAIL:
002A 101 .ASCIC /failed/
64 65 6C 69 61 66 00' 002A
06 002A
0031 102 CREPRC:
0031 103 .ASCIC /CREPRC/
43 52 50 45 52 43 00' 0031
06 0031
0038 104 SETPRV:
0038 105 .ASCIC /SETPRV/
56 52 50 54 45 53 00' 0038
06 0038
003F 106 UNWIND:
003F 107 .ASCIC /UNWIND/
44 4E 49 57 4E 55 00' 003F
06 003F
0046 108 INADR:
0046 109 .LONG NOACCESS,NOACCESS ; page address of noaccess psect
00000000'00000000' 0046
004E 110 PROT:
004E 111 .LONG PRTSC_NA ; protection code for no access psect
00000000' 004E
0052 112 PRVHND_SXV41: ; read only access location
0052 113 CS1:
0052 114 .ASCID \Test !AC service name !AC step !UL failed.\
21 20 74 73 65 54 0000005A'010E0000' 0052
6E 20 65 63 69 76 72 65 73 20 43 41 0060
70 65 74 73 20 43 41 21 20 65 6D 61 006C
2E 64 65 6C 69 61 66 20 4C 55 21 20 0078
0084 115 CS2:
0084 116 .ASCID \Expected !AS = !XL received !AS = !XL\
74 63 65 70 78 45 0000008C'010E0000' 0084
4C 58 21 20 3D 20 53 41 21 20 64 65 0092
41 21 20 64 65 76 69 65 63 65 72 20 009E
4C 58 21 20 3D 20 53 00AA
00B1 117 CS3:
00B1 118 .ASCID \Expected !AS!UB = !XL received !AS!UB = !XL\
74 63 65 70 78 45 000000B9'010E0000' 00B1
20 3D 20 42 55 21 53 41 21 20 64 65 00BF
64 65 76 69 65 63 65 72 20 4C 58 21 00CB
58 21 20 3D 20 42 55 21 53 41 21 20 00D7
4C 00E3
00E4 119 EXP:
00E4 120 .ASCID \status\
00F2 121 NAME_CRE0: ; 0 length string
00F2 122 .ASCID \
00FA 123 NAME_CRE16: ; 16 length string
00FA 124 .ASCID \ABCDEFGHJKLMNOP\
46 45 44 43 42 41 00000102'010E0000' 00FA
50 4F 4E 4D 4C 4B 4A 49 48 47 0108
FF 0112 125 QUOTA_ILLEGAL: ; illegal quota list
0112 126 .BYTE -1
```



```
0113 127 QUOTA_LIST:
01 0113 128 .BYTE PQL$_ASTLM ; minimum quota list
00000002 0114 129 .LONG 2
02 0118 130 .BYTE PQL$_BIOLM
00000002 0119 131 .LONG 2
03 011D 132 .BYTE PQL$_BYTLM
00000400 011E 133 .LONG 1024
04 0122 134 .BYTE PQL$_CPULM
00000000 0123 135 .LONG 0
05 0127 136 .BYTE PQL$_DIOLM
00000002 0128 137 .LONG 2
06 012C 138 .BYTE PQL$_FILLM
00000002 012D 139 .LONG 2
07 0131 140 .BYTE PQL$_PGFLQUOTA
00000100 0132 141 .LONG 256
08 0136 142 .BYTE PQL$_PRCLM
00000000 0137 143 .LONG 0
09 013B 144 .BYTE PQL$_TQELM
00000000 013C 145 .LONG 0
0B 0140 146 .BYTE PQL$_WSDEFAULT
00000064 0141 147 .LONG 100
0A 0145 148 .BYTE PQL$_WSQUOTA
00000064 0146 149 .LONG 100
00 014A 150 .BYTE PQL$_LISTEND
00004000 014B 151 STSFLG_ILLEGAL: ; illegal STS flag bit
014F 152 .LONG ^X4000
00000004 014F 153 STSFLG1: ; inhibit process swapping
0153 154 .LONG 4
52 50 5F 37 31 46 0000015B'010E0000' 0153 155 NAME_CREPRC: ; legal process name
43 4F 0153 156 .ASCID /F17_PROC/
0161
0163 157 GET_LIST:
0004 0163 158 .WORD 4 ; JPI list to get current privs
0400 0165 159 .WORD JPI$_CURPRIV
0000013B 0167 160 .LONG PRIVS
00000000 016B 161 .LONG 0
00000000 016F 162 .LONG 0
0173 163 IMAGE_NAME:
54 55 53 54 41 53 0000017B'010E0000' 0173 164 .ASCID /SATSUT01.EXE/
45 58 45 2E 31 30 0181
```



```
0187 166 ;
0187 167 .SBTTL R/W PSECT
00000000 168 .PSECT RWDATA,RD,WRT,NOEXE,LONG
0000 169 ;
0000 170 IPID:
00000000 0000 171 .LONG 0 ; PID for this process
00000000 0004 172 PID1: .LONG 0 ; PID for target process
00000000 0008 173 CURRENT_TC: .LONG 0 ; ptr to current test case
00000000 000C 174 .ALIGN LONG
00000048 000C 175 REG_SAVE_AREA: .BLKL 15 ; register save area
007480D9 0048 176 MOD_MSG_CODE: .LONG UETP$_SATSMS ; test module message code for putmsg
00000000' 004C 177 TMN_ADDR: .ADDRESS TEST_MOD_NAME
00000019' 0050 178 TMD_ADDR: .ADDRESS TEST_MOD_BEGIN
0054 179 PRVPRT: .BYTE 0 ; protection return byte for SETPRT
00 0054 180 PRIVMASK: .QUAD 0 ; priv. mask
00000000 00000000 0055 181 CHM_CONT: .LONG 0 ; change mode continue address
00000000 005D 182 RETADR: .BLKL 2 ; returned address's from SETPRT
00000069 0061 183 CRE: $CREPRC 0,0,0 ; CREPRC parameter list
0069 184 SET: $SETPRV 0,0,0 ; SETPRV parameter list
00A1 185 UNW: $UNWIND 0,0 ; UNWIND parameter list
00B5 186 REG: .ASCID \register R\
00C1 187 REGNUM: .LONG 0 ; register number
00CF 188 MSGL: .LONG 80 ; buffer desc.
00D3 189 BUF: .ADDRESS BUF
00D7 190 .BLKB 80
00000050 00D7 191 MESSAGEL: .LONG 0 ; message desc.
000000DF' 00DB 192 .ADDRESS
00DF 193 SERV_NAME: .LONG 0 ; service name pointer
0000012F 00DF 194 PRIVS: .QUAD 0 ; privilege storage location
00000000 012F 195 DEPTH: .LONG 0 ; depth storage location for UNWIND
000000DF' 0133 196 WORK: .LONG 0 ; scratch storage location for UNWIND
00000000 0137 197
00000000 013B 198
00000000 0143 199
00000000 0147 200
```



```
00000000 220 .PSECT SATS ACCVIO_1, RD, WRT, NOEXE, PAGE
00000200 0000 221 EMPTY: .BLKB 512 ; reserve a page of space
          0200 222 :
          0200 223 : +
          0200 224 : *****
          0200 225 : *
          0200 226 : * THE ORDER OF STATEMENTS IN THIS PSECT IS CRITICAL. *
          0200 227 : * DO NOT RE-ARRANGE THE VARIABLES. CONSULT SATS *
          0200 228 : * FUNCTIONAL SPECIFICATION FOR A DESCRIPTION OF THE USE *
          0200 229 : * OF THE EMPTY PSECT (AND ITS COMPANION PSECT, NOACCESS). *
          0200 230 : *
          0200 231 : *****
          0200 232 : -
          0200 233 :
000001FF 0200 234 PRVHND_SXV42 = . - 1 ; prvhd arg for SETEXV (last byte in the page)
000001F3 0200 235 = . - 13 ; allow room for string descriptor
          01F3 236 ; type AAAAA_SSSX5 go here:
00000006 01F3 237 .LONG 6 ; string length (will cross psect boundary)
000001FB 01F7 238 .ADDRESS .+4 ; string address
          01FB 239 ; type AAAAA_SSSX3 go here:
000001FC 01FB 240 .BLKB 1 ; low-order byte of string length
          01FC 241 ; type AAAAA_SSSX2 go here:
00000200 01FC 242 .BLKL 1 ; string length
          0200 243 :
          0200 244 :
          0200 245 :
          0200 246 :
          00000000 247 .PSECT SATS ACCVIO_2, RD, WRT, NOEXE, PAGE
00000200 0000 248 NOACCESS: .BLKB 512 ; reserve a page of space
00000000 0200 249 = . - 512 ; return loc ctr to beginning of psect
00000000 0000 250 .ADDRESS EMPTY ; address of accessible string
00000000 0004 251 .ADDRESS EMPTY/^X100 ; address of accessible string
          0008 252 : +
          0008 253 : *** NOTE -- DO NOT CHANGE LOCATION OR SEQUENCE OF ABOVE STATEMENTS!
          0008 254 : *** THIS PSECT (NOACCESS) MUST APPEAR IN MEMORY IMMEDIATELY
          0008 255 : *** FOLLOWING THE EMPTY PSECT. PSECT NAMES AND OPTIONS WILL BE
          0008 256 : *** CHOSEN TO FORCE THE DESIRED PSECT ORDERING.
          0008 257 : -
          0008 258 :
          0008 259 :
          0008 260 :
          0008 261 :
```



```
00000000 263      .PSECT SATSSF18, RD, WRT, EXE, LONG
0000      264      .SBTTL SATSSF18
0000      265      ;++
0000      266      ; FUNCTIONAL DESCRIPTION:
0000      267      ;
0000      268      ;     After performing some initial housekeeping, such as
0000      269      ;     printing the module begin message and acquiring needed privileges,
0000      270      ;     the system services are tested in each of their failure conditions.
0000      271      ;     Detected failures are identified and an error message is printed
0000      272      ;     on the terminal. Upon completion of the test a success or fail
0000      273      ;     message is printed on the terminal.
0000      274      ;
0000      275      ; CALLING SEQUENCE:
0000      276      ;
0000      277      ;     $ RUN SATSSF18 ... (DCL COMMAND)
0000      278      ;
0000      279      ; INPUT PARAMETERS:
0000      280      ;
0000      281      ;     none
0000      282      ;
0000      283      ; IMPLICIT INPUTS:
0000      284      ;
0000      285      ;     none
0000      286      ;
0000      287      ; OUTPUT PARAMETERS:
0000      288      ;
0000      289      ;     none
0000      290      ;
0000      291      ; IMPLICIT OUTPUTS:
0000      292      ;
0000      293      ;     Messages to SYS$OUTPUT are the only output from SATSSF18.
0000      294      ;     They are of the form:
0000      295      ;
0000      296      ;     %UETP-S-SATSMS, TEST MODULE SATSSF18 BEGUN ... (BEGIN MSG)
0000      297      ;     %UETP-S-SATSMS, TEST MODULE SATSSF18 SUCCESSFUL ... (END MSG)
0000      298      ;     %UETP-E-SATSMS, TEST MODULE SATSSF18 FAILED ... (END MSG)
0000      299      ;     %UETP-I-TEXT, ... (VARIABLE INFORMATION ABOUT A TEST MODULE FAILURE)
0000      300      ;
0000      301      ; COMPLETION CODES:
0000      302      ;
0000      303      ;     The SATSSF18 routine terminates with a $EXIT to the
0000      304      ;     operating system with a status code defined by UETP$_SATSMS.
0000      305      ;
0000      306      ; SIDE EFFECTS:
0000      307      ;
0000      308      ;     none
0000      309      ;
0000      310      ; --
0000      311      ;
0000      312      ;
0000      313      ;
0000      314      ; TEST_START SATSSF18                ; let the test begin
```



```

0000 0000
0008'CF 00 DD 0002
0000'CF 02 DF 0006
00000000'GF 00 FB 000C
00000000'GF 00 FB 0013
0009'CF 7F 001A
00000000'GF 01 FB 001E
07FC 30 0025
0050'CF 001F'CF DE 0028
0048'CF 03 00 FO 002F
DD 0036
072B'CF 01 FB 0038
003D
003D
003D
0056

```

STP0:
315
316
317

```

.ENTRY SATSSF18,0
CLRL W^CURRENT_TC
PUSHL #0
PUSHAL W^TPID
CALLS #2,G^SYSSWAKE
CALLS #0,G^SYSSHIBER
PUSHAQ W^TEST_MOD_NAME_D
CALLS #1,G^SYSSSETPRN
BSBW W^MOD_MSG_PRINT
MOVAL W^TEST_MOD_SUCC,W^TMD_ADDR
INSV #SUCCESS,#0,#3,W^MOD_MSG_CODE
PUSHL #0
CALLS #1,W^REG_SAVE

$SETPRT_S INADR=W^INADR, RETADR=W^RETADR, -
PROT=W^PROT, PRVPRT=W^PRVPRT ; set noaccess psect
; ... for no user access

```



```
0056 319 .SBTTL CREPRC TESTS
0056 320 :+
0056 321 :
0056 322 : $CREPRC tests
0056 323 :
0056 324 : test unaccessible PIDADR = page 0 access
0056 325 :
0056 326 :-
0137'CF 0031'CF DE 0056 327 MOVAL W^CREPRC,W^SERV_NAME ; set service name
005D 328 $CREPRC S PIDADR = W^PRVHND_SXV40 ; try it
0081 329 FAIL_CHECK SSS_ACCVIO ; check failure
0735'CF 0C DD 0081
01 FB 0083 PUSHL #SS$ ACCVIO
0088 330 :+
0088 331 :
0088 332 : test unaccessible PIDADR = read-only psect
0088 333 :
0088 334 :-
0088 335 NEXT_TEST
0088
0088 STP1:
0088 MOVL #1,W^CURRENT_TC
008D 008D PUSHL #0
072B'CF 01 FB 008F CALLS #1,W^REG_SAVE
0094 336 $CREPRC S PIDADR = W^PRVHND_SXV41 ; try it
00B8 337 FAIL_CHECK SSS_ACCVIO ; check failure
0735'CF 0C DD 00B8
01 FB 00BA PUSHL #SS$ ACCVIO
00BF 338 :+
00BF 339 :
00BF 340 : test unaccessible PIDADR = noaccess protect
00BF 341 :
00BF 342 :-
00BF 343 NEXT_TEST
00BF
00BF STP2:
00BF MOVL #2,W^CURRENT_TC
00C4 00C4 PUSHL #0
072B'CF 01 FB 00C6 CALLS #1,W^REG_SAVE
00C9 344 $CREPRC S PIDADR = W^PRVHND_SXV42 ; try it
00EF 345 FAIL_CHECK SSS_ACCVIO ; check failure
0735'CF 0C DD 00EF
01 FB 00F1 PUSHL #SS$ ACCVIO
00F6 346 :+
00F6 347 :
00F6 348 : test unaccessible IMAGE = page 0 access
00F6 349 :
00F6 350 :-
00F6 351 NEXT_TEST
00F6
00F6 STP3:
00F6 MOVL #3,W^CURRENT_TC
00FB 00FB PUSHL #0
072B'CF 01 FB 00FD CALLS #1,W^REG_SAVE
0102 352 $CREPRC S IMAGE = W^PRVHND_SXV40 ; try page 0 access
0126 353 FAIL_CHECK SSS_ACCVIO ; check failure
0C DD 0126 PUSHL #SS$ ACCVIO
```



```
0735'CF 01 FB 0128 CALLS #1,W^REG_CHECK
012D 354 :+
012D 355 :
012D 356 : test unaccessible IMAGE = noaccess protection
012D 357 :
012D 358 :-
012D 359 NEXT_TEST
012D
012D STP4:
0008'CF 04 DO 012D MOVL #4,W^CURRENT_TC
00 DD 0132 PUSHL #0
072B'CF 01 FB 0134 CALLS #1,W^REG_SAVE
0139 360 $CREPRC S IMAGE = W^PRVHND_SXV42 ; try noaccess prot
015D 361 FAIL_CHECK SSS_ACCVIO ; check failure
015D
0735'CF 0C DD 015D PUSHL #SS$ ACCVIO
01 FB 015F CALLS #1,W^REG_CHECK
0164 362 :+
0164 363 :
0164 364 : test unaccessible INPUT = page 0 access
0164 365 :
0164 366 :-
0164 367 NEXT_TEST
0164
0164 STP5:
0008'CF 05 DO 0164 MOVL #5,W^CURRENT_TC
00 DD 0169 PUSHL #0
072B'CF 01 FB 0168 CALLS #1,W^REG_SAVE
0170 368 $CREPRC S INPUT = W^PRVHND_SXV40 ; try it
0194 369 FAIL_CHECK SSS_ACCVIO ; check failure
0194
0735'CF 0C DD 0194 PUSHL #SS$ ACCVIO
01 FB 0196 CALLS #1,W^REG_CHECK
019B 370 :+
019B 371 :
019B 372 : test unaccessible INPUT = noaccess protect
019B 373 :
019B 374 :-
019B 375 NEXT_TEST
019B
019B STP6:
0008'CF 06 DO 019B MOVL #6,W^CURRENT_TC
00 DD 01A0 PUSHL #0
072B'CF 01 FB 01A2 CALLS #1,W^REG_SAVE
01A7 376 $CREPRC S INPUT = W^PRVHND_SXV42 ; try it
01CB 377 FAIL_CHECK SSS_ACCVIO
01CB
0735'CF 0C DD 01CB PUSHL #SS$ ACCVIO
01 FB 01CD CALLS #1,W^REG_CHECK
01D2 378 :+
01D2 379 :
01D2 380 : test unaccessible OUTPUT = page 0 access
01D2 381 :
01D2 382 :-
01D2 383 NEXT_TEST
01D2
01D2 STP7:
0008'CF 07 DO 01D2 MOVL #7,W^CURRENT_TC
00 DD 01D7 PUSHL #0
072B'CF 01 FB 01D9 CALLS #1,W^REG_SAVE
```



```
0735'CF  OC  DD  01DE 384  $CREPRC S OUTPUT = W^PRVHND_SXV40 ; try it
          01  FB  0202 385  FAIL_CHECK SSS_ACCVIO ; check failure
          01  FB  0202 385  PUSHL #SS$ ACCVIO
          01  FB  0204 386  CALLS #1,W^REG_CHECK
          01  FB  0209 386  :+
          01  FB  0209 387  :
          01  FB  0209 388  : test unaccessible OUTPUT = noaccess protect
          01  FB  0209 389  :
          01  FB  0209 390  :-
          01  FB  0209 391  NEXT_TEST
          01  FB  0209 391  STP8:
          01  FB  0209 392  MOVL #8,W^CURRENT_TC
          01  FB  020E 392  PUSHL #0
          01  FB  0210 392  CALLS #1,W^REG_SAVE
          01  FB  0215 392  $CREPRC S OUTPUT = W^PRVHND_SXV42 ; try it
          01  FB  0239 393  FAIL_CHECK SSS_ACCVIO ; check failure
          01  FB  0239 393  PUSHL #SS$ ACCVIO
          01  FB  023B 393  CALLS #1,W^REG_CHECK
          01  FB  0240 394  :+
          01  FB  0240 395  :
          01  FB  0240 396  : test unaccessible ERROR = page 0 access
          01  FB  0240 397  :
          01  FB  0240 398  :-
          01  FB  0240 399  NEXT_TEST
          01  FB  0240 399  STP9:
          01  FB  0240 400  MOVL #9,W^CURRENT_TC
          01  FB  0245 400  PUSHL #0
          01  FB  0247 400  CALLS #1,W^REG_SAVE
          01  FB  024C 400  $CREPRC S ERROR = W^PRVHND_SXV40 ; try it
          01  FB  0270 401  FAIL_CHECK SSS_ACCVIO ; check failure
          01  FB  0270 401  PUSHL #SS$ ACCVIO
          01  FB  0272 401  CALLS #1,W^REG_CHECK
          01  FB  0277 402  :+
          01  FB  0277 403  :
          01  FB  0277 404  : test unaccessible ERROR = noaccess protect
          01  FB  0277 405  :
          01  FB  0277 406  :-
          01  FB  0277 407  NEXT_TEST
          01  FB  0277 407  STP10:
          01  FB  0277 408  MOVL #10,W^CURRENT_TC
          01  FB  027C 408  PUSHL #0
          01  FB  027E 408  CALLS #1,W^REG_SAVE
          01  FB  0283 408  $CREPRC S ERROR = W^PRVHND_SXV42 ; try it
          01  FB  02A7 409  FAIL_CHECK SSS_ACCVIO ; check failure
          01  FB  02A7 409  PUSHL #SS$ ACCVIO
          01  FB  02A9 409  CALLS #1,W^REG_CHECK
          01  FB  02AE 410  :+
          01  FB  02AE 411  :
          01  FB  02AE 412  : test unaccessible PRVADR = page 0 access
          01  FB  02AE 413  :
          01  FB  02AE 414  :-
          01  FB  02AE 415  NEXT_TEST
          01  FB  02AE 415  STP11:
```



```
0008'CF 0B DO 02AE          MOVL    #11,W^CURRENT_TC
00      00 DD 02B3          PUSHL   #0
072B'CF 01 FB 02B5          CALLS   #1,W^REG_SAVE
02BA      416      $CREPRC S PRVADR = W^PRVHND_SXV40      ; try it
02DE      417      FAIL_CHECK SSS_ACCVIO                  ; check failure
0735'CF 0C DD 02DE          PUSHL   #SS$ ACCVIO
01      01 FB 02E0          CALLS   #1,W^REG_CHECK
02E5      418      ;+
02E5      419      ; test unaccessable PRVADR = noaccess protect
02E5      420      ;
02E5      421      ;
02E5      422      ; -
02E5      423      ;
02E5      423      NEXT_TEST
02E5      STP12:
0008'CF 0C DO 02E5          MOVL    #12,W^CURRENT_TC
00      00 DD 02EA          PUSHL   #0
072B'CF 01 FB 02EC          CALLS   #1,W^REG_SAVE
02F1      424      $CREPRC S PRVADR = W^PRVHND_SXV42      ; try it
0315      425      FAIL_CHECK SSS_ACCVIO                  ; check failure
0735'CF 0C DD 0315          PUSHL   #SS$ ACCVIO
01      01 FB 0317          CALLS   #1,W^REG_CHECK
031C      426      ;+
031C      427      ; test unaccessable QUOTA = page 0 access
031C      428      ;
031C      429      ;
031C      430      ; -
031C      431      ;
031C      431      NEXT_TEST
031C      STP13:
0008'CF 0D DO 031C          MOVL    #13,W^CURRENT_TC
00      00 DD 0321          PUSHL   #0
072B'CF 01 FB 0323          CALLS   #1,W^REG_SAVE
0328      432      $CREPRC S QUOTA = W^PRVHND_SXV40      ; try it
034C      433      FAIL_CHECK SSS_ACCVIO                  ; check failure
0735'CF 0C DD 034C          PUSHL   #SS$ ACCVIO
01      01 FB 034E          CALLS   #1,W^REG_CHECK
0353      434      ;+
0353      435      ; test unaccessable QUOTA = noaccess protect
0353      436      ;
0353      437      ;
0353      438      ; -
0353      439      ;
0353      439      NEXT_TEST
0353      STP14:
0008'CF 0E DO 0353          MOVL    #14,W^CURRENT_TC
00      00 DD 0358          PUSHL   #0
072B'CF 01 FB 035A          CALLS   #1,W^REG_SAVE
01FF'CF 01 90 035F          MOVB    #PQL$ ASTLM,W^PRVHND_SXV42 ; set an initial quota in the first
0364      440      $CREPRC S QUOTA = W^PRVHND_SXV42      ; try it
0388      441      FAIL_CHECK SSS_ACCVIO                  ; check failure
0735'CF 0C DD 0388          PUSHL   #SS$ ACCVIO
01      01 FB 038A          CALLS   #1,W^REG_CHECK
038F      443      ;+
038F      444      ; test unaccessable PRCNAM = page 0 access
038F      445      ;
038F      446      ;
```



```
038F 447 :-  
038F 448 NEXT_TEST  
038F  
038F STP15:  
0008'CF 0F DO 038F MOVL #15,W^CURRENT_TC  
00 DD 0394 PUSHL #0  
072B'CF 01 FB 0396 CALLS #1,W^REG_SAVE  
0398 449 $CREPRC S PRCNAM = W^PRV^ND_SXV40 ; try it  
03BF 450 FAIL_CHECK SS$_ACCVIO ; check failure  
03BF PUSHL #SS$_ACCVIO  
0735'CF 0C DD 03C1 CALLS #1,W^REG_CHECK  
01 FB 03C6  
03C6 451 :+  
03C6 452 : test unaccessable PRCNAM = noaccess protect  
03C6 453 :  
03C6 454 :  
03C6 455 :-  
03C6 456 NEXT_TEST  
03C6  
03C6 STP16:  
0008'CF 10 DO 03C6 MOVL #16,W^CURRENT_TC  
00 DD 03CB PUSHL #0  
072B'CF 01 FB 03CD CALLS #1,W^REG_SAVE  
03D2 457 $CREPRC S PRCNAM = W^PRV^ND_SXV42 ; try it  
03F6 458 FAIL_CHECK SS$_ACCVIO ; check failure  
03F6 PUSHL #SS$_ACCVIO  
0735'CF 0C DD 03F8 CALLS #1,W^REG_CHECK  
01 FB 03FD  
03FD 459 :+  
03FD 460 : test PRCNAM = 16 length string  
03FD 461 :  
03FD 462 :  
03FD 463 :-  
03FD 464 NEXT_TEST  
03FD  
03FD STP17:  
0008'CF 11 DO 03FD MOVL #17,W^CURRENT_TC  
00 DD 0402 PUSHL #0  
072B'CF 01 FB 0404 CALLS #1,W^REG_SAVE  
0409 465 $CREPRC S PRCNAM = W^NAME_CRE16 ; try it  
042D 466 FAIL_CHECK SS$_IVLOGNAM ; check failure  
042D PUSHL #SS$_IVLOGNAM  
00000154 8F DD 0433 CALLS #1,W^REG_CHECK  
0735'CF 01 FB 0438  
0438 467 :+  
0438 468 : test SS$_IVQUOTAL  
0438 469 :  
0438 470 :  
0438 471 :-  
0438 472 NEXT_TEST  
0438  
0438 STP18:  
0008'CF 12 DO 0438 MOVL #18,W^CURRENT_TC  
00 DD 043D PUSHL #0  
072B'CF 01 FB 043F CALLS #1,W^REG_SAVE  
0444 473 $CREPRC S QUOTA = W^QUOTA_ILLEGAL ; try it  
0468 474 FAIL_CHECK SS$_IVQUOTAL ; check failure  
0468 PUSHL #SS$_IVQUOTAL  
00000164 8F DD 046E CALLS #1,W^REG_CHECK  
0735'CF 01 FB 0473  
0473 475 :+
```



```
0473 476 : test SSS_IVSTSFLG
0473 477 :
0473 478 :
0473 479 :-
0473 480 NEXT_TEST
0473
0473 STP19:
0008'CF 13 DO 0473 MOVL #19,W^CURRENT_TC
00 DD 0478 PUSHL #0
072B'CF 01 FB 047A CALLS #1,W^REG_SAVE
047F 481 $CREPRC S STSFLG = W^STSFLG_ILLEGAL ; try it
04A3 482 FAIL_CHECK SSS_IVSTSFLG ; check failure
0000017C 8F DD 04A3 PUSHL #SS$ IVSTSFLG
0735'CF 01 FB 04A9 CALLS #1,W^REG_CHECK
04AE 483 :+
04AE 484 :
04AE 485 : test SSS_NOPRIV
04AE 486 :
04AE 487 :-
04AE 488 NEXT_TEST
04AE
04AE STP20:
0008'CF 14 DO 04AE MOVL #20,W^CURRENT_TC
00 DD 04B3 PUSHL #0
072B'CF 01 FB 04B5 CALLS #1,W^REG_SAVE
04BA 489 $CREPRC S STSFLG = W^STSFLG1 ; try it
04DE 490 FAIL_CHECK SSS_NOPRIV ; check failure
04DE
04E0
0735'CF 24 DD 04DE PUSHL #SS$ NOPRIV
01 FB 04E0 CALLS #1,W^REG_CHECK
04E5 491 :+
04E5 492 :
04E5 493 : test SSS_DUPLNAM
04E5 494 :
04E5 495 :-
04E5 496 NEXT_TEST
04E5
04E5 STP21:
0008'CF 15 DO 04E5 MOVL #21,W^CURRENT_TC
00 DD 04EA PUSHL #0
072B'CF 01 FB 04EC CALLS #1,W^REG_SAVE
04F1 497 $CREPRC S QUOTA=W^QUOTA [IST,- ; make a legal process
04F1 498 PRCNAM = W^NAME CREPRC,-
04F1 499 IMAGE=W^IMAGE_NAME,-
04F1 500 PIDADR=W^PID1
051B 501 FAIL_CHECK SSS_NORMAL ; try S with IMAGE param.
051B 501 PUSHL #SS$ NORMAL ; check success
051D
0735'CF 01 DD 051B
01 FB 051D CALLS #1,W^REG_CHECK
0522 502 $CREPRC S PRCNAM = W^NAME_CREPRC ; try an illegal one
0546 503 FAIL_CHECK SSS_DUPLNAM ; check failure
0546
00000094 8F DD 0546
0735'CF 01 FB 054C
0551 504 $WAKE_S PIDADR = W^PID1 ; cause process termination
```



```

055E 506 .SBTTL SETPRV TESTS
055E 507 :+
055E 508 :
055E 509 $SETPRV tests
055E 510 :
055E 511 test unaccessible PRVADR = page 0 access
055E 512 :
055E 513 :-
055E 514 NEXT_TEST
055E
055E STP22:
055E          MOVL    #22,W^CURRENT_TC
0563          PUSHL   #0
0565          CALLS   #1,W^REG_SAVE
056A          MOVAL   W^SETPRV,W^SERV_NAME          ; set service name
0571          $SETPRV S PRVADR = W^PRVHND_SXV40      ; try it
0582          FAIL_CHECK SSS_ACCVIO                  ; check failure
0582          PUSHL   #SSS_ACCVIO
0584          CALLS   #1,W^REG_CHECK
0589
0589 :+
0589 518 :
0589 519 :
0589 520 test unaccessible PRVADR = noaccess protect
0589 521 :
0589 522 :-
0589 523 NEXT_TEST
0589
0589 STP23:
0589          MOVL    #23,W^CURRENT_TC
058E          PUSHL   #0
0590          CALLS   #1,W^REG_SAVE
0595          $SETPRV S PRVADR = W^PRVHND_SXV42      ; try it
05A6          FAIL_CHECK SSS_ACCVIO                  ; check the failure
05A6          PUSHL   #SSS_ACCVIO
05A8          CALLS   #1,W^REG_CHECK
05AD
05AD :+
05AD 526 :
05AD 527 :
05AD 528 test unaccessible PRVPRV = page 0 access
05AD 529 :
05AD 530 :-
05AD 531 NEXT_TEST
05AD
05AD STP24:
05AD          MOVL    #24,W^CURRENT_TC
05B2          PUSHL   #0
05B4          CALLS   #1,W^REG_SAVE
05B9          $SETPRV S PRVPRV = W^PRVHND_SXV40      ; try it
05CA          FAIL_CHECK SSS_ACCVIO                  ; check failure
05CA          PUSHL   #SSS_ACCVIO
05CC          CALLS   #1,W^REG_CHECK
05D1
05D1 :+
05D1 534 :
05D1 535 :
05D1 536 test unaccessible PRVPRV = read-only psect
05D1 537 :
05D1 538 :-
05D1 539 NEXT_TEST
05D1
05D1 STP25:

```



```
0008'CF 19 DO 05D1          MOVL #25,W^CURRENT_TC
00 DD 05D6          PUSHL #0
072B'CF 01 FB 05D8          CALLS #1,W^REG_SAVE
05DD 540 $SETPRV S PRVPRV = W^PRV^ND_SXV41 ; try it
05EE 541 FAIL_CHECK SSS_ACCVIO ; check failure
0C DD 05EE          PUSHL #SS$ ACCVIO
0735'CF 01 FB 05F0          CALLS #1,W^REG_CHECK
05F5 542 :-+
05F5 543 :-+
05F5 544 :-+ test unaccessable PRVPRV = noaccess protect
05F5 545 :-+
05F5 546 :-+
05F5 547 :-+
05F5          NEXT_TEST
05F5          STP26:
0008'CF 1A DO 05F5          MOVL #26,W^CURRENT_TC
00 DD 05FA          PUSHL #0
072B'CF 01 FB 05FC          CALLS #1,W^REG_SAVE
0601 548 $SETPRV S PRVPRV = W^PRV^ND_SXV42 ; try it
0612 549 FAIL_CHECK SSS_ACCVIO ; check failure
0C DD 0612          PUSHL #SS$ ACCVIO
0735'CF 01 FB 0614          CALLS #1,W^REG_CHECK
```



```
0619 551 .SBTTL UNWIND TESTS
0619 552 :+
0619 553 :
0619 554 : $UNWIND tests
0619 555 :
0619 556 : test SS$_NOSIGNAL
0619 557 :
0619 558 :-
0619 559 NEXT_TEST

0619 STP27:
0619 MOVL #27,W^CURRENT_TC
061E PUSHL #0
0620 CALLS #1,W^REG_SAVE
0625 MOVAL W^UNWIND,W^SERV_NAME ; set service name
062C MOVL #1,W^DEPTH ; set the depth
0631 $UNWIND S DEPADR = W^DEPTH ; try it
063E FAIL_CHECK SS$_NOSIGNAL ; check failure
063E PUSHL #SS$_NOSIGNAL
0644 CALLS #1,W^REG_CHECK

0649 564 :+
0649 565 :
0649 566 : test SS$_INSFRAME
0649 567 :
0649 568 :-
0649 569 NEXT_TEST

0649 STP28:
0649 MOVL #28,W^CURRENT_TC
064E PUSHL #0
0650 CALLS #1,W^REG_SAVE
0655 INCL W^DEPTH ; set the unwind depth
0659 MOVL SP,W^WORK ; remember the stack pointer
065E CALLS #0,B^10$ ; put a call frame on the stack
0662 570 10$:
0662 .WORD 0
0664 571 MOVAL B^20$(FP) ; set the handler address
0668 572 CLRL SF$$_SAVE_FP(SP) ; put a stop in the stack unwind cha
066B 573 20$:
066B CHMU #0 ; cause an exception
066D 574
066D 575 .WORD ^M<R2>
066F 576 MOVL B^CHF$$_SIGARGLST(AP),R2 ; get signal array address
0673 577 PUSHL #0 ; push a dummy parameter
0675 578 CALLS #1,W^REG_SAVE ; save a reg snapshot
067A 579 $UNWIND S DEPADR = W^DEPTH,NEWPC = B^30$ ; do it
0688 580 CLRL @SF$$_SAVE_FP(FP) ; disable the handler for error msg
068B 581 MOVL W^WORK,SP ; reset the stack pointer
0690 582 MOVL SP,FP ; reset the FP
0693 583 FAIL_CHECK SS$_INSFRAME ; check failure
0693 PUSHL #SS$_INSFRAME
0699 584 CALLS #1,W^REG_CHECK

069E 588 30$:
069E 589 :+
069E 590 :
069E 591 : test SS$_UNWINDING
069E 592 :
069E 593 :-
```



```
069E 594 NEXT_TEST
069E
069E STP29:
0008'CF 1D D0 069E MOVL #29,W^CURRENT_TC
00 00 DD 06A3 PUSHL #0
072B'CF 01 FB 06A5 CALLS #1,W^REG_SAVE
0143'CF D7 06AA 595 DECL W^DEPTH ; set to a legal depth
B2'AF 00 FB 06AE 596 CALLS #0,B^10$ ; put a call frame on the stack
06B2 597 10$:
06B2 598 .WORD 0
6D BA'AF 0000 06B4 599 MOVAL B^20$,(FP) ; set the handler address
00 00 BF 06B8 600 CHMU #0 ; cause an exception
06BA 601 20$:
0004 06BA 602 .WORD ^M<R2>
52 04 AC D0 06BC 603 MOVL CHF$ _SIGARGLST(AP),R2 ; get the signal array address
00 00 DD 06C0 604 PUSHL #0 ; push a dummy parameter
072B'CF 01 FB 06C2 605 CALLS #1,W^REG_SAVE ; save a reg snapshot
06C7 606 $UNWIND_S DEPADR = W^DEPTH,NEWPC = B^30$ ; do it
04 A2 00000920 8F D1 06D5 607 CMPL #SS$_UNWIND,B^CHF$ _SIG_NAME(R2) ; are we unwinding?
11 13 06DD 608 BEQL 15$ ; br if yes
0C BD D4 06DF 609 CLRL @SF$ _SAVE_FP(FP) ; disable the handler
06E2 610 FAIL_CHECK SS$_NORMAL ; check failure
01 DD 06E2 PUSHL #SS$_NORMAL
0735'CF 01 FB 06E4 CALLS #1,W^REG_CHECK
0C BD CE AF DE 06E9 611 MOVAL B^20$,@SF$ _SAVE_FP(FP) ; enable the handler
13 11 06EE 612 BRB 17$ ; continue in common
06F0 613 15$:
0C BD D4 06F0 614 CLRL @SF$ _SAVE_FP(FP) ; disable the handler
06F3 615 FAIL_CHECK SS$_UNWINDING ; check failure
00000928 8F DD 06F3 PUSHL #SS$_UNWINDING
0735'CF 01 FB 06F9 CALLS #1,W^REG_CHECK
0C BD B9 AF DE 06FE 616 MOVAL B^20$,@SF$ _SAVE_FP(FP) ; enable the handler
0703 617 17$:
04 0703 618 RET ; giver heck
0704 619 30$:
0704 620 :+
0704 621 :-
0704 622 : Testing SS$_ACCVIO will not be done because of the hostile results
0704 623 : that can occur from intentionally corrupting the STACK.
0704 624 :-
0704 625 :-
0704 626 TEST_END ; thats all folks
0050'CF DD 0704 PUSHL W^TMD_ADDR
004C'CF DD 0708 PUSHL W^TMN_ADDR
02 DD 070C PUSHL #2
0048'CF DD 070E PUSHL W^MOD_MSG_CODE
00000000'GF 04 FB 0712 CALLS #SST1,G^LIB$SIGNAL
0048'CF 01 1C 01 FO 0719 INSV #1,#ST$V_INHIB_MSG,#1,W^MOD_MSG_CODE
00000000'GF 01 DD 0720 PUSHL W^MOD_MSG_CODE
FB 0724 CALLS #1,G^SYS$EXIT
```



```
072B 628 .SBTTL REG_SAVE
072B 629 :++
072B 630 : FUNCTIONAL DESCRIPTION:
072B 631 : Subroutine to save R2-R11 in the register save location.
072B 632 :
072B 633 : CALLING SEQUENCE:
072B 634 : PUSH  #0 ; save a dummy parameter
072B 635 : CALLS  #1,W^REG_SAVE ; save R2-R11
072B 636 :
072B 637 : INPUT PARAMETERS:
072B 638 : NONE
072B 639 :
072B 640 : OUTPUT PARAMETERS:
072B 641 : NONE
072B 642 :
072B 643 :--
072B 644 :
072B 645 REG_SAVE:
072B 646 .WORD  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
000C'CF 14 AD 28 OFFC 072D 647 MOV  #4*10,^X14(FP),W^REG_SAVE_AREA ; save the registers in the program
0734 648 RET
0735 649 .SBTTL REG_CHECK
0735 650 :++
0735 651 : FUNCTIONAL DESCRIPTION:
0735 652 : Subroutine to test R0 & R2-R11 for proper content after a service
0735 653 : execution. A snapshot is taken by the REG_SAVE routine at the
0735 654 : beginning of each step and this routine is executed after the
0735 655 : services have been executed.
0735 656 :
0735 657 : CALLING SEQUENCE:
0735 658 : PUSH  #SS$ XXXXXX ; push expected R0 contents
0735 659 : CALLS  #1,W^REG_CHECK ; execute this routine
0735 660 :
0735 661 : INPUT PARAMETERS:
0735 662 : expected R0 contents on the stack
0735 663 :
0735 664 : OUTPUT PARAMETERS:
0735 665 : possible error messages printed using $PUTMSG
0735 666 :
0735 667 :--
0735 668 :
0735 669 REG_CHECK:
0735 670 .WORD  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
50 04 AC D1 0737 671 CMPL  4(AP),R0 ; is this the right fail code?
073B 672 BEQL  10$ ; br if yes
073D 673 PUSH  R0 ; push received data
073F 674 PUSH  4(AP) ; push expected data
00E4'CF DF 0742 675 PUSHAL W^EXP ; push the string variable
077D'CF 03 FB 0746 676 CALLS  #3,W^PRINT_FAIL ; print the error message
074B 677 10$:
000C'CF 14 AD 28 29 074B 678 CMPC3  #4*10,^X14(FP),W^REG_SAVE_AREA ; check all but R0
56 53 0000000C'8F C3 0752 679 BEQL  20$ ; br if O.K.
0754 680 SUBL3  #REG_SAVE_AREA,R3,R6 ; calculate the register number
075C 681 DIVL2  #4,R6
075F 682 ADDB3  #^X2,R6,W^REGNUM ; put it in the string
0765 683 BICL2  #3,R1 ; backup to register boundry
0768 684 BICL2  #3,R3
```



```
00D3'CF DD 076B 685      PUSHL  W^REGNUM      ; push register number
        61 DD 076F 686      PUSHL  (R1)        ; push received data
        63 DD 0771 687      PUSHL  (R3)        ; push expected data
00C1'CF DF 0773 688      PUSHAL  W^REG        ; set string pntr param.
077D'CF 04 FB 0777 689      CALLS  #4,W^PRINT_FAIL ; print the error message
        077C 690 20$:
04      077C 691      RET
        077D 692      .SBTTL  PRINT_FAIL
        077D 693      :++
        077D 694      : FUNCTIONAL DESCRIPTION:
        077D 695      : Subroutine to report failures using $PUTMSG
        077D 696      :
        077D 697      : CALLING SEQUENCE:
        077D 698      : Mode #1      PUSHL EXPECTED Mode #2      PUSHL REG_NUMBER
        077D 699      :              PUSHL RECEIVED          PUSHL EXPECTED
        077D 700      :              FUSHAL STRING_VAR        PUSHL RECEIVED
        077D 701      :              CALLS #3,W^PRINT_FAIL    PUSHAL STRING_VAR
        077D 702      :                                  CALLS #4,W^PRINT_FAIL
        077D 703      :
        077D 704      : INPUT PARAMETERS:
        077D 705      : listed above
        077D 706      :
        077D 707      : OUTPUT PARAMETERS:
        077D 708      : an error message is printed using $PUTMSG
        077D 709      :
        077D 710      :--
        077D 711      :
        077D 712      PRINT_FAIL:
003C    077D 713      .WORD  ^M<R2,R3,R4,R5>
        077F 714      $FAO_S  W^CS1,W^MESSAGEL,W^MSGSL,#TEST_MOD_NAME,W^SERV_NAME,W^CURRENT_TC
        07A0 715      PUTMSG  <#UETPS_TEXT,#1,#MESSAGEL>      ; print the message
04      6C    91 07B5 716      CMPB  (AP),#4              ; is this a register message?
        21    13 07B8 717      BEQL  10$                  ; br if yes
        25    11 07BA 718      $FAO_S W^CS2,W^MESSAGEL,W^MSGSL,4(AP),8(AP),4(AP),12(AP)
        07D9 719      BRB  20$                          ; goto output message
        07DB 720 10$:
        07DB 721      $FAO_S  W^CS3,W^MESSAGEL,W^MSGSL,4(AP),16(AP),8(AP),4(AP),16(AP),12(AP)
        0800 722 20$:
        0800 723      PUTMSG  <#UETPS_TEXT,#1,#MESSAGEL>      ; print the message
0050'CF 002A'CF DE 0815 724      MOVAL  W^TEST_MOD_FAIL,W^TMD_ADDR ; set failure message address
0048'CF 03 00 02 FO 081C 725      INSV  #ERROR,#0,#3,W^MOD_MSG_CODE ; set severity code
        04 0823 726      RET
```



```
0824 728 .SBTTL MOD_MSG_PRINT
0824 729 MOD_MSG_PRINT:
0824 730 :
0824 731 : *****
0824 732 : *
0824 733 : * PRINTS THE TEST MODULE BEGUN/SUCCESSFUL/FAILED MESSAGES *
0824 734 : * (USING THE PUTMSG MACRO). *
0824 735 : *
0824 736 : *****
0824 737 :
05 0824 738 PUTMSG <W^MOD_MSG_CODE,#2,W^TMN_ADDR,W^TMD_ADDR> : PRINT MSG
0839 739 RSB ; ... AND RETURN TO CALLER
083A 740 :
083A 741 .SBTTL CHMRTN
083A 742 CHMRTN:
083A 743 : *****
083A 744 : *
083A 745 : * CHANGE MODE ROUTINE. THIS ROUTINE GETS CONTROL WHENEVER *
083A 746 : * A CMKRNL, CMEXEC, OR CMSUP SYSTEM SERVICE IS ISSUED *
083A 747 : * BY THE MODE MACRO ('TO' OPTION). IT MERELY DOES *
083A 748 : * A JUMP INDIRECT ON A FIELD SET UP BY MODE. IT HAS *
083A 749 : * THE EFFECT OF RETURNING TO THE END OF THE MODE *
083A 750 : * MACRO EXPANSION. *
083A 751 : *
083A 752 : *****
083A 753 :
0000005D'FF 0000 083A 754 .WORD 0 ; ENTRY MASK
17 083C 755 JMP @CHM_CONT ; RETURN TO MODE MACRO IN NEW MODE
0842 756 :
0842 757 : * RET INSTR WILL BE ISSUED IN EXPANSION OF 'MODE FROM, ....' MACRO
0842 758 :
0842 759 .END SATSSF18
```


SATSSF18
Symbol table

B 4
- SATS SYSTEM SERVICE TESTS (FAILING S. 16-SEP-1984 01:42:11 VAX/VMS Macro V04-00
5-SEP-1984 04:22:29 [UETP.SRC]SATSSF18.MAR;1

Page 23
(2)

\$\$ARGS	= 00000002			PQL\$-WSDEFAULT	= 0000000B		
\$\$T1	= 00000004			PQL\$-WSQUOTA	= 0000000A		
\$\$T2	= 00000009			PRINT_FAIL	0000077D	R	06
BUF	000000DF	R	03	PRIVMASK	00000055	R	03
CHF\$-SIGARGLST	= 00000004			PRIVS	0000013B	R	03
CHF\$-SIG_NAME	= 00000004			PROT	0000004E	R	02
CHMRTN	0000083A	R	06	PRT\$C_NA	*****	X	02
CHM_CONT	0000005D	R	03	PRVHND-SXV40	= 00000001		
CRE	00000069	R	03	PRVHND-SXV41	00000052	R	02
CREPRC	00000031	R	02	PRVHND-SXV42	= 000001FF	R	04
CREPRC\$-BASPRI	= 00000024			PRVPRT	00000054	R	03
CREPRC\$-ERROR	= 00000014			QUOTA_ILLEGAL	00000112	R	02
CREPRC\$-IMAGE	= 00000008			QUOTA_LIST	00000113	R	02
CREPRC\$-INPUT	= 0000000C			REG	000000C1	R	03
CREPRC\$-ITMLST	= 00000034			REGNUM	000000D3	R	03
CREPRC\$-MBXUNT	= 0000002C			REG_CHECK	00000735	R	06
CREPRC\$-NARGS	= 0000000D			REG_SAVE	0000072B	R	06
CREPRC\$-OUTPUT	= 00000010			REG_SAVE_AREA	0000000C	R	03
CREPRC\$-PIDADR	= 00000004			RETADR	00000061	R	03
CREPRC\$-PRCNAM	= 00000020			SATSSF18	00000000	RG	06
CREPRC\$-PRVADR	= 00000018			SERV_NAME	00000137	R	03
CREPRC\$-QUOTA	= 0000001C			SET	000000A1	R	03
CREPRC\$-STSFLG	= 00000030			SETPRV	00000038	R	02
CREPRC\$-UIC	= 00000028			SETPRV\$-ENBFLG	= 00000004		
CS1	00000052	R	02	SETPRV\$-NARGS	= 00000004		
CS2	00000084	R	02	SETPRV\$-PRMFLG	= 0000000C		
CS3	000000B1	R	02	SETPRV\$-PRVADR	= 00000008		
CURRENT_TC	00000008	R	03	SETPRV\$-PRVPRV	= 00000010		
DEPTH	00000143	R	03	SEVERE	= 00000004		
EMPTY	00000000	R	04	SF\$-SAVE_FP	= 0000000C		
ERROR	= 00000002			SHR\$-SHRDEF	= 00000001		
EXP	000000E4	R	02	SHR\$-TEXT	= 00001130		
GET_LIST	00000163	R	02	SS\$-ACCVIO	= 0000000C		
IMAGE_NAME	00000173	R	02	SS\$-DUPLNAM	= 00000094		
INADR	00000046	R	02	SS\$-INSFRAME	= 0000012C		
INFO	= 00000003			SS\$-IVLOGNAM	= 00000154		
JPI\$-CURPRIV	= 00000400			SS\$-IVQUOTAL	= 00000164		
LIB\$-SIGNAL	*****	X	06	SS\$-IVSTSFLG	= 0000017C		
MESSAGEL	0000012F	R	03	SS\$-NOPRIV	= 00000024		
MOD_MSG_CODE	00000048	R	03	SS\$-NORMAL	= 00000001		
MOD_MSG_PRINT	00000824	R	06	SS\$-NOSIGNAL	= 00000900		
MSG	000000D7	R	03	SS\$-UNWIND	= 00000920		
NAME_CREO	000000F2	R	02	SS\$-UNWINDING	= 00000928		
NAME_CRE16	000000FA	R	02	STEP	= 0000001D		
NAME_CREPRC	00000153	R	02	STP0	0000003D	R	06
NOACCESS	00000000	R	05	STP1	00000088	R	06
PID1	00000004	R	03	STP10	00000277	R	06
PQL\$-ASTLM	= 00000001			STP11	000002AE	R	06
PQL\$-BIOLM	= 00000002			STP12	000002E5	R	06
PQL\$-BYTLM	= 00000003			STP13	0000031C	R	06
PQL\$-CPULM	= 00000004			STP14	00000353	R	06
PQL\$-DIOLM	= 00000005			STP15	0000038F	R	06
PQL\$-FILLM	= 00000006			STP16	000003C6	R	06
PQL\$-LISTEND	= 00000000			STP17	000003FD	R	06
PQL\$-PGFLQUOTA	= 00000007			STP18	00000438	R	06
PQL\$-PRCLM	= 00000008			STP19	00000473	R	06
PQL\$-TQELM	= 00000009			STP2	000000BF	R	06

SATSSF18
Symbol table

- SATS SYSTEM SERVICE TESTS (FAILING S. 16-SEP-1984 01:42:11 VAX/VMS Macro V04-00
5-SEP-1984 04:22:29 [UETP.SRC]SATSSF18.MAR;1

Page 24
(2)

STP20	000004AE	R	06
STP21	000004E5	R	06
STP22	0000055E	R	06
STP23	00000589	R	06
STP24	000005AD	R	06
STP25	000005D1	R	06
STP26	000005F5	R	06
STP27	00000619	R	06
STP28	00000649	R	06
STP29	0000069E	R	06
STP3	000000F6	R	06
STP4	0000012D	R	06
STP5	00000164	R	06
STP6	0000019B	R	06
STP7	000001D2	R	06
STP8	00000209	R	06
STP9	00000240	R	06
STSSV_INHIB_MSG	= 0000001C		
STSFLG1	0000014F	R	02
STSFLG_ILLEGAL	0000014B	R	02
SUCCESS	= 00000001		
SYSSCREPRC	*****	GX	06
SYSEXIT	*****	GX	06
SYSSFAO	*****	X	06
SYSSHIBER	*****	GX	06
SYSSSETPRN	*****	GX	06
SYSSSETPRT	*****	GX	06
SYSSSETPRV	*****	GX	06
SYSSUNWIND	*****	GX	06
SYSSWAKE	*****	GX	06
TEST_MOD_BEGIN	00000019	R	02
TEST_MOD_FAIL	0000002A	R	02
TEST_MOD_NAME	00000000	R	02
TEST_MOD_NAME_D	00000009	R	02
TEST_MOD_SUCC	0000001F	R	02
TMD_ADDR	00000050	R	03
TMN_ADDR	0000004C	R	03
TPID	00000000	R	03
UETPS_SATSMS	= 007480D9		
UETPS_TEXT	= 00741133		
UNW	000000B5	R	03
UNWIND	0000003F	R	02
UNWINDS_DEPADR	= 00000004		
UNWINDS_NARGS	= 00000002		
UNWINDS_NEWPC	= 00000008		
WARNING	= 00000000		
WORK	00000147	R	03

SAT
V04

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00C00000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
RODATA	00000187 (391.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
RWDATA	00000148 (331.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
SATS_ACCVIO_1	00000200 (512.)	04 (4.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
SATS_ACCVIO_2	00000200 (512.)	05 (5.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
SATSSF18	00000842 (2114.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	37	00:00:00.09	00:00:00.32
Command processing	138	00:00:00.69	00:00:03.02
Pass 1	403	00:00:15.45	00:00:36.53
Symbol table sort	0	00:00:01.41	00:00:02.68
Pass 2	232	00:00:03.69	00:00:09.78
Symbol table output	27	00:00:00.16	00:00:00.26
Psect synopsis output	6	00:00:00.04	00:00:00.11
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	845	00:00:21.54	00:00:52.71

The working set limit was 900 pages.

97103 bytes (190 pages) of virtual memory were used to buffer the intermediate code.

There were 50 pages of symbol table space allocated to hold 939 non-local and 12 local symbols.

759 source lines were read in Pass 1, producing 32 object records in Pass 2.

48 pages of virtual memory were used to define 42 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-\$255\$DUA28:[UETP.OBJ]UETP.MLB;1	10
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	29
TOTALS (all libraries)	39

1154 GETS were required to define 39 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:SATSSF18/OBJ=OBJ\$:SATSSF18 MSRC\$:SATSSF18/UPDATE=(ENH\$:SATSSF18)+EXECMLS/LIB+LIB\$:UETP/LIB

0410 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY